HttpClient

一、 HttpClient 简介

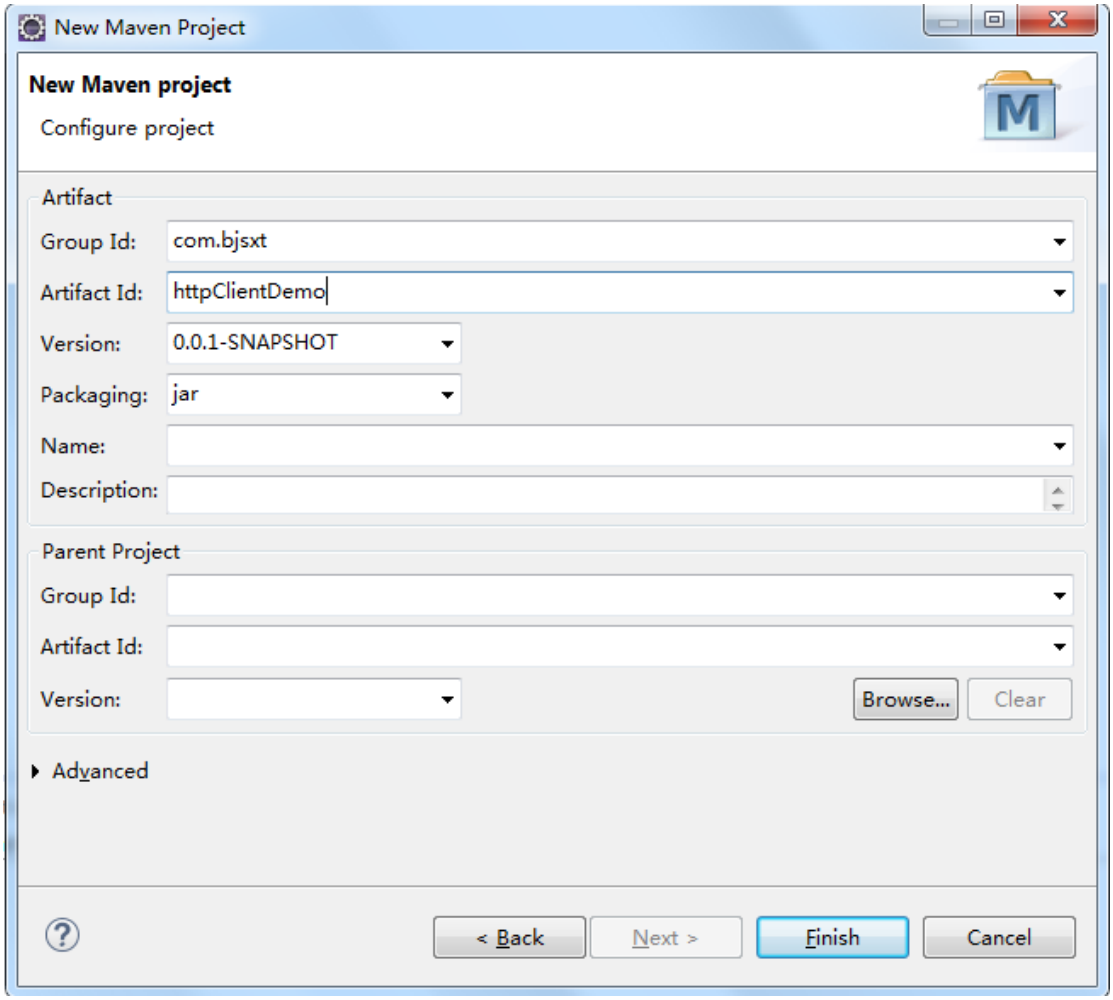
HttpClient 是 Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

HTTP 协议可能是现在 Internet 上使用得最多、最重要的协议了，越来越多的 Java 应用程序需要直接通过 HTTP 协议来访问网络资源。虽然在 JDK 的 java net 包中已经提供了访问 HTTP 协议的基本功能，但是对于大部分应用程序来说，JDK 库本身提供的功能还不够丰富和灵活。

二、 HttpClient 应用

1 发送 GET 请求不带参数

1.1创建项目



1.2 修改 POM 文件添加 HttpClient 坐标

```
    <project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
ce"
     xsi:schemaLocation="http://maven.apache.org/P
OM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
     <modelVersion>4.0.0</modelVersion>
     <groupId>com.bjsxt</groupId>
     <artifactId>httpClientDemo</artifactId>
     <version>0.0.1-SNAPSHOT</version>

     <dependencies>
        <!--
https://mvnrepository.com/artifact/org.apache.http
components/httpclient -->
        <dependency>

   <groupId>org.apache.httpcomponents</groupId>
           <artifactId>httpclient</artifactId>
           <version>4.3.5</version>
```

```
        </dependency>
      </dependencies>
    </project>
```

1.3编写测试代码

```
    public class HttpClientTest {


      public static void main(String[] args) throws
Exception {
        HttpClientTest.doGet();
      }


      /**
       * Get 请求不带参数
       * @throws Exception
       * @throws ClientProtocolException
       */
      public static  void doGet() throws  Exception{
        //常见一个 HttpClient 对象
        CloseableHttpClient client =
HttpClients.createDefault();
        //创建 Get 请求对象。在请求中输入 url
```

```
        HttpGet get = new
HttpGet("http://www.baidu.com");
        //发送请求，并返回响应
        CloseableHttpResponse res =
client.execute(get);
        //处理响应
        //获取响应的状态码
        int code =
res.getStatusLine().getStatusCode();
        System.out.println(code);
        //获取响应的内容
        HttpEntity entity = res.getEntity();
        String content =
EntityUtils.toString(entity,"utf-8");
        System.out.println(content);
        //关闭连接
        client.close();
      }
    }
```

2 发送 GET 请求带参数

```
    /**
```

```
     * Get 请求带参数
     * @throws Exception
     */
    public static void doGetParam() throws
Exception{
        CloseableHttpClient client =
HttpClients.createDefault();
        //创建一个封装 URI 的对象。在该对象中可以给定请
求参数
        URIBuilder  bui = new
URIBuilder("https://www.sogou.com/web");
        bui.addParameter("query", "西游记");
        //创建一个 Get 请求对象
        HttpGet get = new HttpGet(bui.build());

        //发送请求，并返回响应
        CloseableHttpResponse res =
client.execute(get);
        //处理响应
        //获取响应的状态码
        int code =
res.getStatusLine().getStatusCode();
```

```
        System.out.println(code);
        //获取响应的内容
        HttpEntity entity = res.getEntity();
        String content =
EntityUtils.toString(entity,"utf-8");
        System.out.println(content);
        //关闭连接
        client.close();
    }
```

3 发送 POST 请求不带参数

```
    /**
      * 发送 POST 请求不带参数
      */
    public static void doPostTest()throws
Exception{
        CloseableHttpClient client =
HttpClients.createDefault();
        HttpPost post = new
HttpPost("http://localhost:8080/test/post");
        CloseableHttpResponse res =
```

```
client.execute(post);


        //处理响应
        //获取响应的状态码
        int code =
res.getStatusLine().getStatusCode();
        System.out.println(code);
        //获取响应的内容
        HttpEntity entity = res.getEntity();
        String content =
EntityUtils.toString(entity,"utf-8");
        System.out.println(content);
        //关闭连接
        client.close();
     }
```

4 发送 POST 请求带参数

```
     /**
      * 发送 POST 请求带参数
      */
     public static void doPostParamTest()throws
Exception{
```

```
        CloseableHttpClient client =
HttpClients.createDefault();
        HttpPost post = new
HttpPost("http://localhost:8080/test/post/param");
        //给定参数
        List<BasicNameValuePair> list = new
ArrayList<>();
        list.add(new BasicNameValuePair("name", "张
三丰"));
        list.add(new BasicNameValuePair("pwd",
"zhangsanfeng"));
        //将参数做字符串的转换
        StringEntity entity = new
UrlEncodedFormEntity(list,"utf-8");
        //向请求中绑定参数
        post.setEntity(entity);

        //处理响应
        CloseableHttpResponse res =
client.execute(post);
        //获取响应的状态码
        int code =
```

```
res.getStatusLine().getStatusCode();
        System.out.println(code);
        //获取响应的内容
        HttpEntity en = res.getEntity();
        String content =
EntityUtils.toString(en,"utf-8");
        System.out.println(content);
        //关闭连接
        client.close();
     }
```

5　在 POST 请求的参数中传递 JSON 格式数据

```
    /**
       * 发送 POST 请求带 JSON 格式参数
       */
     public static void doPostParamJsonTest()throws
Exception{
        CloseableHttpClient client =
HttpClients.createDefault();
        HttpPost post = new
HttpPost("http://localhost:8080/test/post/param/js
on");
```

```
        String json ="{\"name\":\"张三丰\",\"pwd\":\"zhangsanfeng\"}";
        StringEntity entity = new StringEntity(json, ContentType.APPLICATION_JSON);
        //向请求中绑定参数
        post.setEntity(entity);


        //处理响应
        CloseableHttpResponse res = client.execute(post);
        //获取响应的状态码
        int code = res.getStatusLine().getStatusCode();
        System.out.println(code);
        //获取响应的内容
        HttpEntity en = res.getEntity();
        String content = EntityUtils.toString(en,"utf-8");
        System.out.println(content);
        //关闭连接
        client.close();
    }
```

6 HttpClient 自定义工具类的使用

6.1编写工具类

```
    public class HttpClientUtil {


     public static String doGet(String url,
Map<String, String> param) {


        // 创建 Httpclient 对象
        CloseableHttpClient httpclient =
HttpClients.createDefault();


        String resultString = "";
        CloseableHttpResponse response = null;
        try {
           // 创建 uri
           URIBuilder builder = new URIBuilder(url);
           if (param != null) {
              for (String key : param.keySet()) {
                 builder.addParameter(key,
param.get(key));
              }
```

```
            }
            URI uri = builder.build();

            // 创建 http GET 请求
            HttpGet httpGet = new HttpGet(uri);

            // 执行请求
            response = httpclient.execute(httpGet);
            // 判断返回状态是否为 200
            if
(response.getStatusLine().getStatusCode() == 200) {
                resultString =
EntityUtils.toString(response.getEntity(),
"UTF-8");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (response != null) {
                    response.close();
                }
```

```
            httpclient.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return resultString;
}

public static String doGet(String url) {
    return doGet(url, null);
}

public static String doPost(String url,
Map<String, String> param) {
    // 创建 Httpclient 对象
    CloseableHttpClient httpClient =
HttpClients.createDefault();
    CloseableHttpResponse response = null;
    String resultString = "";
    try {
        // 创建 Http Post 请求
        HttpPost httpPost = new HttpPost(url);
```

```
            // 创建参数列表
            if (param != null) {
                List<NameValuePair> paramList = new
ArrayList<>();
                for (String key : param.keySet()) {
                    paramList.add(new
BasicNameValuePair(key, param.get(key)));
                }
                // 模拟表单
                UrlEncodedFormEntity entity = new
UrlEncodedFormEntity(paramList,"utf-8");
                httpPost.setEntity(entity);
            }
            // 执行 http 请求
            response = httpClient.execute(httpPost);
            resultString =
EntityUtils.toString(response.getEntity(),
"utf-8");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
```

```
            response.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    return resultString;
}

public static String doPost(String url) {
    return doPost(url, null);
}

public static String doPostJson(String url,
String json) {
    // 创建Httpclient对象
    CloseableHttpClient httpClient =
HttpClients.createDefault();
    CloseableHttpResponse response = null;
    String resultString = "";
    try {
```

```
            // 创建 Http Post 请求
            HttpPost httpPost = new HttpPost(url);
            // 创建请求内容
            StringEntity entity = new
StringEntity(json, ContentType.APPLICATION_JSON);
            httpPost.setEntity(entity);
            // 执行 http 请求
            response = httpClient.execute(httpPost);
            resultString =
EntityUtils.toString(response.getEntity(),
"utf-8");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                response.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

```
        return resultString;
    }
  }
```

6.2测试工具类

```
  /**
    * 测试 HttpClient 工具类
    */
    public static void httpClientUtilTest(){
      String url =
"http://localhost:8080/test/post/param";
      Map<String, String> param = new HashMap<>();
      param.put("name", "李四");
      param.put("pwd", "lisi");
      String result = HttpClientUtil.doPost(url,
param);
      System.out.println(result);
    }
```

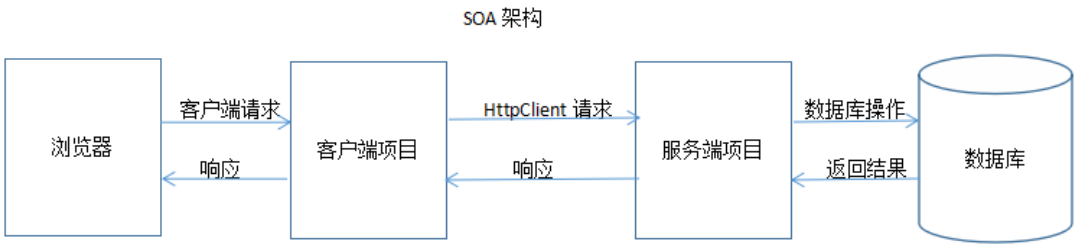
三、 实战案例

1 需求

1） 采用 SOA 架构项目

2） 使用 HttpClient 调用服务
3） 完成用户的添加与查询

2 项目架构



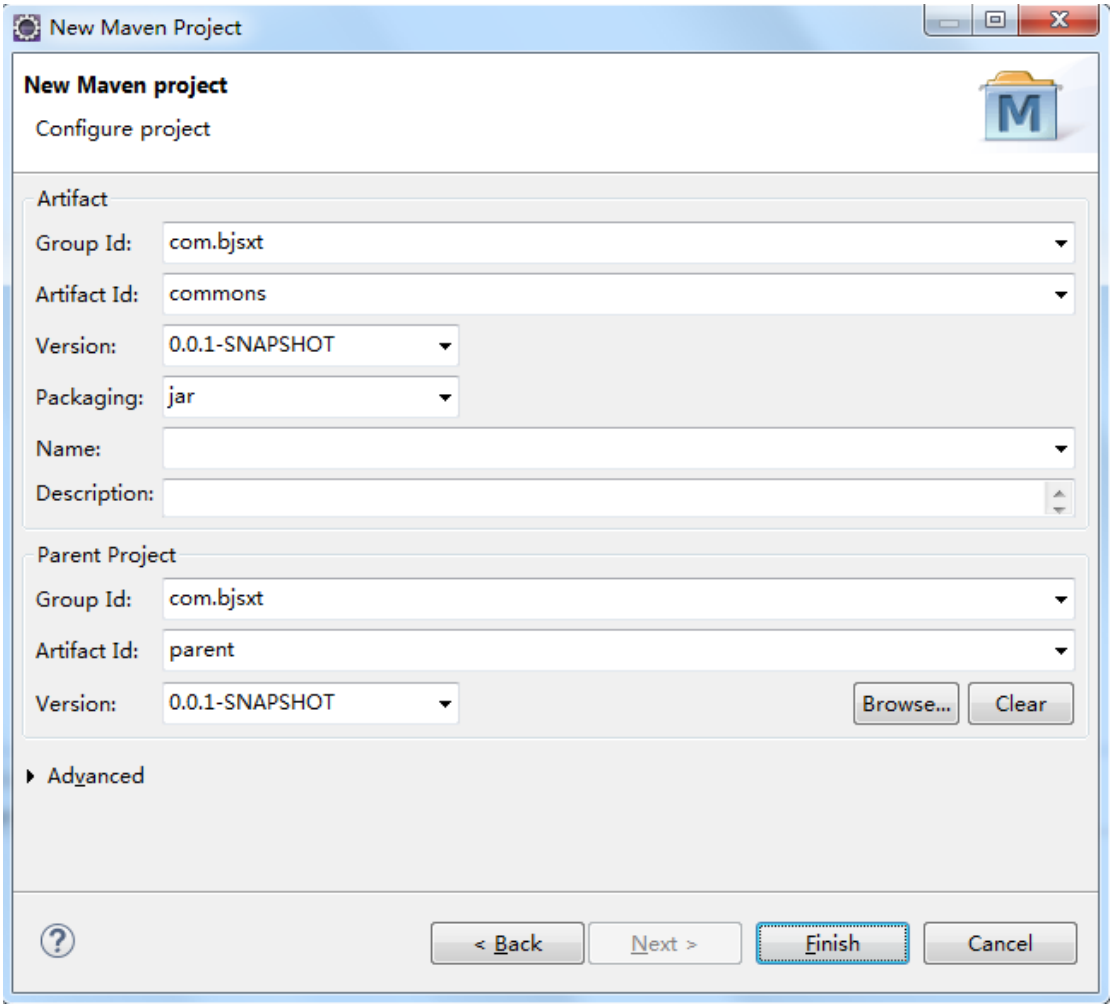
3 表结构

```
CREATE TABLE `users` (
  `userid` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(30) DEFAULT NULL,
  `userage` int(11) DEFAULT NULL,
  PRIMARY KEY (`userid`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
```

4 创建项目

4.1创建 commons 项目

4.1.1创建项目



4.1.2需改 POM 文件

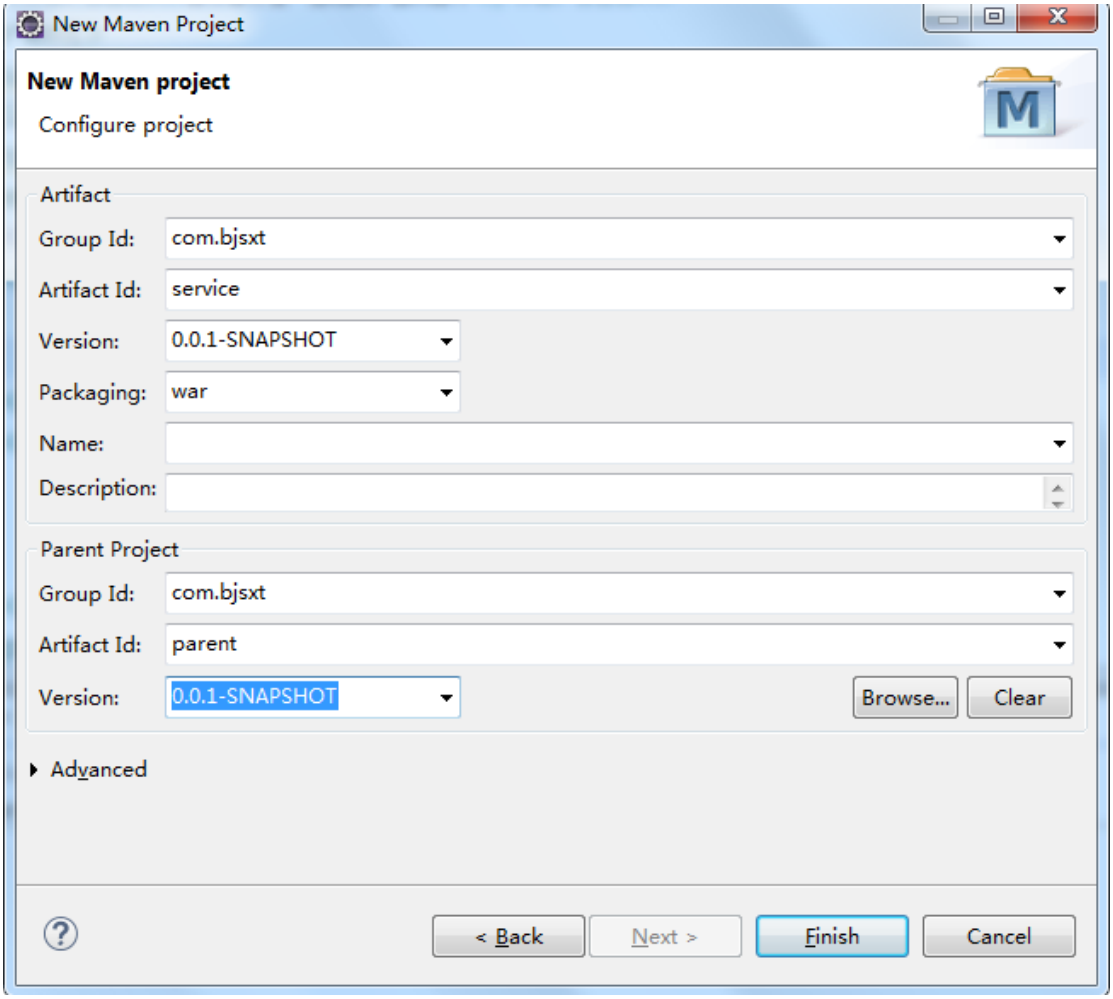
```
    <project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
ce"
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
      <groupId>com.bjsxt</groupId>
      <artifactId>parent</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </parent>
    <groupId>com.bjsxt</groupId>
    <artifactId>commons</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
      <!-- Jackson Json 处理工具包 -->
      <dependency>

  <groupId>com.fasterxml.jackson.core</groupId>

  <artifactId>jackson-databind</artifactId>
      </dependency>
      <dependency>
```

```
<groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.3.5</version>
    </dependency>
  </dependencies>
 </project>
```

4.2 创建 service 项目

4.2.1 创建项目



4.2.2修改 POM 文件

```
    <project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
ce"
      xsi:schemaLocation="http://maven.apache.org/P
OM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
         <groupId>com.bjsxt</groupId>
         <artifactId>parent</artifactId>
         <version>0.0.1-SNAPSHOT</version>
      </parent>
      <groupId>com.bjsxt</groupId>
      <artifactId>service</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>war</packaging>

      <dependencies>
         <dependency>
            <groupId>com.bjsxt</groupId>
```

```
        <artifactId>commons</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <!-- 单元测试 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
        </dependency>
        <!-- 日志处理 -->
        <dependency>
            <groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>
        </dependency>
        <!-- Mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
```

```
<artifactId>mybatis-spring</artifactId>
        </dependency>
        <!-- MySql -->
        <dependency>
           <groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- 连接池 -->
        <dependency>
           <groupId>com.alibaba</groupId>
           <artifactId>druid</artifactId>
        </dependency>
        <!-- Spring -->
        <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
```

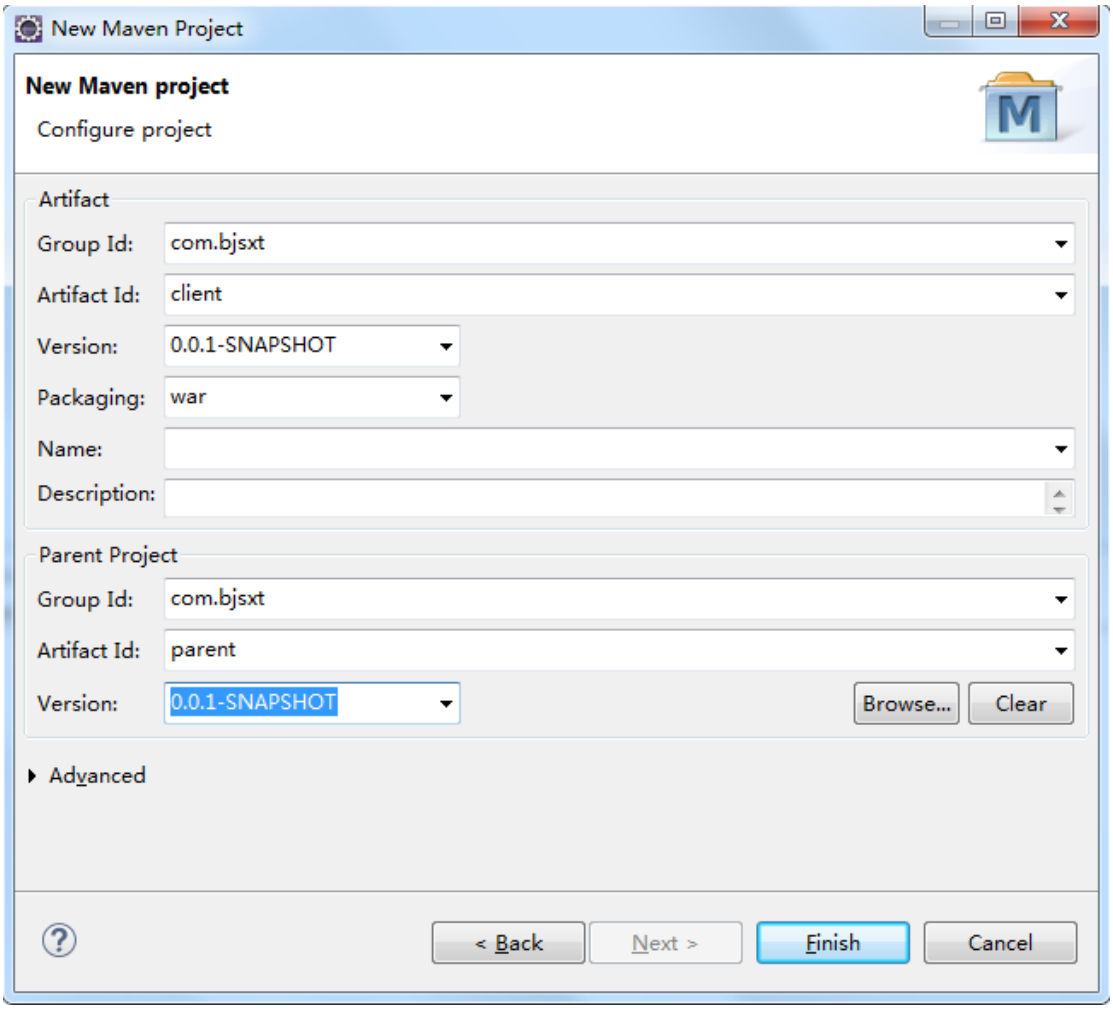
```
<groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
```

```
            <artifactId>servlet-api</artifactId>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
      <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
              <include>**/*.xml</include>
            </includes>
        </resource>
        <resource>

  <directory>src/main/resources</directory>
            <includes>
              <include>**/*.xml</include>
              <include>**/*.properties</include>
            </includes>
        </resource>
      </resources>
```

```
        <!-- tomcat 插件，由于子项目不一定每个都是 web
项目，所以该插件只是声明，并未开启 -->
          <plugins>
            <!-- 配置 Tomcat 插件 -->
            <plugin>

  <groupId>org.apache.tomcat.maven</groupId>

  <artifactId>tomcat7-maven-plugin</artifactId>
              <configuration>
                <path>/</path>
                <port>8080</port>
              </configuration>
            </plugin>
          </plugins>
      </build>
    </project>
```

4.3创建 client 项目

4.3.1创建项目



4.3.2修改 POM 文件

```
    <project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
ce"
xsi:schemaLocation="http://maven.apache.org/POM/4.
```

```
0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
        <groupId>com.bjsxt</groupId>
        <artifactId>parent</artifactId>
        <version>0.0.1-SNAPSHOT</version>
      </parent>
      <groupId>com.bjsxt</groupId>
      <artifactId>client</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>war</packaging>
      <dependencies>
        <dependency>
           <groupId>com.bjsxt</groupId>
           <artifactId>commons</artifactId>
           <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <!-- 单元测试 -->
           <dependency>
              <groupId>junit</groupId>
              <artifactId>junit</artifactId>
           </dependency>
```

```
        <!-- 日志处理 -->
        <dependency>
          <groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>
        </dependency>

        <!-- Spring -->
        <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework</groupId>
          <artifactId>spring-beans</artifactId>
        </dependency>
        <dependency>

<groupId>org.springframework</groupId>
```

```
<artifactId>spring-webmvc</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <!-- tomcat 插件，由于子项目不一定每个都是 web
项目，所以该插件只是声明，并未开启 -->
```

```
        <plugins>
            <!-- 配置 Tomcat 插件 -->
            <plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <path>/</path>
                    <port>8081</port>
                </configuration>
            </plugin>
        </plugins>
    </build>
  </project>
```

5 添加用户

5.1 Client

5.1.1addUser.jsp

```
    <%@ page language="java" contentType="text/html;
charset=UTF-8"
```

```
        pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
      <form action="/user/addUser" method="post">
        用户姓名：<input type="text"
name="username"/><br/>
        用户年龄：<input type="text"
name="userage"/><br/>
        <input type="submit" value="OKOK"/>
      </form>
    </body>
    </html>
```

5.1.2UserService

```
    @Override
      public void addUser(Users user) {
         String json = JsonUtils.objectToJson(user);
         String code =
HttpClientUtil.doPostJson("http://localhost:8080/u
ser/insertUser", json);
         Map<String, Integer> map =
JsonUtils.jsonToPojo(code, Map.class);
         Integer var = map.get("code");
         if(var == 500){
            System.out.println("出错了");
         }else{
            System.out.println("添加成功");
         }


      }
```

5.1.3UserController

```
    @Controller
    @RequestMapping("/user")
    public class UserController {
```

```
    @Autowired
    private UserService userService;

    /**
     * 添加用户
     */
    @RequestMapping("/addUser")
    public String addUser(Users user){
        this.userService.addUser(user);
        return "ok";
    }
}
```

5.2 Service

5.2.1 UserController

```
@Controller
@RequestMapping("/user")
public class UserController {

    @Autowired
```

```
    private UserService userService;


    @RequestMapping("/insertUser")
    @ResponseBody
    public Object insertUser(@RequestBody Users
user){
        Map<String, Integer> map = new HashMap<>();
        try{
            this.userService.insertUser(user);
            map.put("code", 200);
        }catch(Exception e){
            e.printStackTrace();
            map.put("code", 500);
        }
        return map;
    }
  }
```

5.2.2UserService

```
  @Service
  public class UserServiceImpl implements
```

```
UserService {


      @Autowired
      private UserMapper userMapper;


      @Override
      public void insertUser(Users user) {
         this.userMapper.insertUser(user);
      }


    }
```

6 查询用户

6.1 Client

6.1.1 UserController

```
      /**
       * 查询全部用户
       */
      @RequestMapping("/findUser")
      public String findUserAll(Model model){
         List<Users> list =
this.userService.findUserAll();
```

```
        model.addAttribute("list", list);
        return "showUser";
    }
```

6.1.2UserService

```
    @Override
    public List<Users> findUserAll() {
        String var =
HttpClientUtil.doPost("http://localhost:8080/user/
selectUserAll");
        List<Users> list = JsonUtils.jsonToList(var,
Users.class);
        return list;
    }
```

6.2Service

6.2.1UserController

```
    @RequestMapping("/selectUserAll")
    @ResponseBody
    public Object selectUserAll(){
        return this.userService.selectUserAll();
    }
```

6.2.2UserService

```
@Override
public List<Users> selectUserAll() {
    return this.userMapper.selectUserAll();
}
```